CSRD (2009) 23: 185–193
DOI 10.1007/s00450-009-0094-4

SPECIAL ISSUE PAPER

Generalized utility metrics for supercomputers

Erich Strohmaier

Published online: 19 May 2009

Abstract The problem of ranking the utility of supercomputer systems arises frequently in situations such as procurements and other types of evaluations of architectures. It is also central for any general ranking of supercomputers such as the Top500. Rankings of computer systems have traditionally solely focused on performance aspects. In recent years restrictions due to power and space requirements of large supercomputers have become very noticeable, which has increased the importance of including these factors in generalized rankings. In this paper we present an overview of the current practice for utility metrics and analyze their shortcomings. We then present and discuss in detail a new concept for a parameterized utility metric for supercomputers, which is based on effective performance, available memory size, actual power consumption, and (if desired) the floor space required for supercomputers. This metric is designed and proposed for augmenting the current Top500 ranking.

Keywords Computer performance · Utility metrics · Power efficiency · High performance computing market analysis

1 Introduction

Power consumption of computer systems in general and supercomputers in particular has been rising constantly over the last decades and has become a major concern for facility requirements and cost of operation. A convenient definition of what a supercomputer is has been provided for 16 years by the Top500 project [1, 2]. The Top500 ranks installed computer systems by their Linpack performance [3] and the first 500 are called "supercomputer". The current ranking methodology does not make any attempt to include any other system features in the determination of the rank of a system. Features of interest include available memory size, power consumption, space requirements, and others.

Attempts have been made to overcome this shortcoming. Most recently the Green500 is sorting the systems of the Top500 by power-efficiency and is republishing them [4]. Unfortunately, power-efficiency as any efficiency or density can not be used to rank objects by "size", which leads to substantial flaws in the Green500 approach. To overcome these shortcomings and to provide alternative ranking procedures, we have researched how to construct generalized utility metrics for supercomputers, which include the effects of power consumption and other features. We analyzed the basic requirements for any utility metrics to be useful in our context and derived a general expandable parameterized form for such a metric.

To further restricted the possible form of such metrics we analyze different answers to the simple question: *"When is my new supercomputer twice as useful as my old one?"* We then characterize several system upgrade scenarios with the new utility metric to show that our requirements for a useful metric are met and our intuitive understanding of these scenarios is properly reflected.

As an illustration of the effects of using the new utility metric we take the subset of Top500 systems with power consumption and memory data available and compare new ranks with the original Top500 rank. We also analyze the quantitative impact of individual correction factors.

This work was supported by the ASCR Office in the DOE Office of Science under contract number DE-AC02-05CH11231.

E. Strohmaier (✉)
Future Technologies Group,
Lawrence Berkeley National Laboratory,
One Cyclotron Road,
Berkeley, CA 94720, USA
e-mail: estrohmaier@lbl.gov

2 Current practice for ranking computers

The current practice for ranking supercomputers (and other computer systems) is dominated by performance based metrics. Prominent examples are the Top500 Supercomputer list, which ranks actual installed systems, or benchmark suites such as HPCC [5], or SPEC [6], which rank computer system models. In procurement situations the access to actual cost data allows ranking competing offers with econometric measures such as life-time costs and cost-performance. New approaches for metrics, which include other aspects of computers, include the Green500 [7], where actual installed high performance computing (HPC) systems from the Top500 are re-ranked by power-efficiency (performance/power). Another interesting concept developed by Tsugio Makimoto is the "Figure of Merit" for nomadic devices, which inspired the work presented here [8].

2.1 Performance and efficiency metrics

It is natural to compare the progress in computational capabilities with metrics such as floating point performance or bandwidth. These metrics are "extensive" quantities as they typically grow with the "size" or capability of a computer system. They can be used to rank systems by capability and track our increasing computing capabilities over time [2].

The enabling technologies, however, are often better characterized by "intensive" quantities such as efficiencies or densities. Intensive quantities are typically constructed by dividing two extensive quantities. Important examples would be the *Byte/Flop* ratio, the *power efficiency = performance/power consumption*, and the *power density = power/space*. These intensive quantities capture important features of our basic technologies and can be used to compare and track the progress (or decline) of these technologies. However, they cannot be used to rank or track computer systems by *size*!

2.2 Performance rankings

Computer systems have been ranked by their achieved performance for a long time. While there have been rankings based on fictitious performance numbers or theoretical peak performance numbers, most of these rankings focus on measured performances. Benchmarks used for these rankings vary in complexity greatly from simply synthetic tests such as STREAM [9] to full (scientific) applications as often used in focused performance studies [10, 11]. The benchmarks and problem sizes used in these rankings often have to be adapted or replaced to reflect the changing usage of computer systems.

A prominent example is the Top500 List of Supercomputers [1]. Twice a year it ranks the 500 highest performing installed computer systems based on the Linpack benchmark [3]. This procedure provides a self-adapting definition of what can be considered a supercomputer at a given point in time and has proven its value over time. It allows following and analyzing the HPC market despite rapidly changing architectures and performance levels. The main purposes of the Linpack benchmark in this context are: i) Separating real existing systems from nonfunctional ones and ii) Providing a first order correction to peak performance. Practice has shown that it indeed does pose a serious first stability test for new supercomputer systems. However, its ability to reflect performance of scientific application workloads has degraded substantially over time. Important features which allow the usage of Linpack in the Top500 are a scalable problem definition and the implicit property that only full usage of available memory allows to maximize performance.

The HPCC benchmark suite is the most prominent attempt to overcome the later shortcoming of the Linpack benchmark [5]. The suite consists of seven different kernel benchmark (including Linpack), selected to test different performance aspects of large scale systems. It lists performance measurements of various systems and system sizes, but does not provide listings of actual installed systems. Problem sizes are again scalable.

The SPEC effort is a similar industry driven effort for providing benchmark suites. It initially focused on workstation class systems and has since then branched out into different class of computer systems. Problem sizes of benchmarks are fixed. To deal with increasing system capabilities, SPEC has adopted the model of defining new benchmark suites every several years. Recently SPEC has started to look seriously into the question of power consumption of computer systems and how to properly measure it.

2.3 Procurement situations

Procurement situations are different as they include access to actual price and cost data. The main criteria for selecting a specific computer system might be performance or functionality, but in many situations capital life-time cost, operational life-time cost, and cost-performance play an important or decisive role. Accounting for the cost of the power consumption of a computer system requires appropriate definitions and measurements, but is otherwise straight forward. Even space and general infrastructure requirements can be evaluated in terms of cost.

The difficult aspect of procurements is the construction of representative benchmark suites and the evaluation of the provided performance results [13]. Once this is achieved a ranking based on *(life-time performance)/(life-time cost)* can be compiled.

Unfortunately this very desirable econometric methodology cannot be adopted for general supercomputer rankings

such as the Top500. It would require accurate data about the price of a system and the utility and space cost for each site. All of these data are highly individualized, vary greatly between different sites and systems, and cannot be obtained with reasonable effort.

2.4 Green500

In a recent effort to promote awareness of power consumption of supercomputers, the Green500 project began publishing re-rankings of the Top500 systems based on power-efficiency [4,7]. Unfortunately, this metric is *intensive* and thus can not be used to rank computer systems by size. It is a very appropriate metric to compare technologies, but its usage for ranking actual installed computer systems has serious flaws.

Power efficiency as an intensive quantity does not allow sorting systems by any size reflecting capability. If the Green500 re-ranking would not limit itself to the subset of Top500 systems it is very likely that small but very power efficient computing devices such as laptops or even cellphones would be at the top of the resulting ranking. This does not represent any capability of performing large calculations, power-efficient or not.

Power-efficiency is a feature of technology. Ranking x systems of type A and y systems of type B, will therefore in almost all cases lead to ranking all system of type A ahead or behind all systems of type B. Any information about size or capability is lost and will not be reflected in the ranking.

The Green500 extrapolates power consumption of large system from small systems. Thus power is a linear function of system size. Linpack is a slightly sub-linear function of system size due to decreasing parallel efficiency. Dividing a sub-linear function by a linear function leads to a monotone decreasing function with size. Therefore, in the Green500 smaller systems will always be ranked higher than larger systems of the same type. Within a class of systems with similar technologies smaller system will always rank higher than larger ones. This surely is the opposite of what most people would require from a listing of supercomputers.

These flaws are a direct consequence of using an intensive metric such as power-efficiency for ranking actual systems and thus cannot be overcome. Intensive metrics can only be used to rank technologies, for which they are very useful.

2.5 Makimoto's figure of merit for the nomadic age

In 1994 Tsugio Makimoto presented a "Figure of Merit" for capturing the usefulness of mobile (nomadic) tools [8]. The requirements of high "intelligence" (aka computing capability), low power, low space, and low cost for such devices lead him to define:

$$Figure\ of\ Merit = \frac{Intelligence}{Size \times Cost \times Power}\,. \tag{1}$$

This figure of merit (*FM*) allows tracking the progress of e.g. hand-calculators over two decades during which their *FM* increases by roughly 11 orders of magnitude [8]. As *FM* is composed by dividing one intensive quantity by three intensive quantities it greatly favors smaller systems over larger ones, which clearly was its intention. While the metric itself is not useful for ranking supercomputers, it provides one of the basic ideas on how to construct a generalized utility metric for such a purpose. It multiplies all desired quantities (computing capability) and divides them by undesired quantities (Size, Cost, and Space). In the next section we will expand on this construction principle and define a family of metrics suitable for ranking supercomputer systems.

3 Utility metrics for supercomputers

To construct a utility metric (*UM*) for supercomputers we have to answer the question, how to quantify the utility a system provides to us, as function of its basic characteristics such as:

- Achieved Performance,
- Usable Memory,
- Effective Power Consumption,
- Space Requirements,

and potentially others such as disk space. We do not include cost here for reasons discussed in Sect. 2. In essence this problem is equivalent to the condensed and simplified question: *When is my new supercomputer twice as useful as my old one?*

One should keep in mind, that the choice of performance metric is essential for arriving at a meaningful *UM*. Any system feature of interest such as processor, memory, or I/O should have direct influence on performance values assigned to a system. Considering these system features only as additional term in the *UM* itself can in general not lead to a satisfactory end-result by itself.

3.1 Basic requirements for utility metrics

The relative importance of system features is different for different users, which implies, that our factors should affect the utility metric based on weights adjustable for individual needs. Specific usage scenarios for computer systems can then be used to derive restricting relations between these weight parameters.

A key requirement for a useful utility metric is that relative ranks based on it do not depend on scales or units cho-

sen for any contributing factor x. In other words: $UM(a \times x) = f(a) \times UM(x)$ and $f(a)$ has to be a strongly increasing, monotone function for all features x. This requirement rules out any simple additive metric. The simplest functional form to fulfill this requirement is indeed multiplicative like Makimoto's Figure of Merit. Including adjustable weight factors as exponents we arrive at a weighted product of all factors as simplest form for our utility metric (UM):

$$UM(x_i) = \prod x_i^{\alpha_i} . \tag{2}$$

The weights for beneficial factors will be positive, while weights for detrimental factors will be negative.

As performance is the most important factor for typical supercomputers, we like to construct our utility such that it serves as correction to ranking with the extensive factor performance alone. This is easiest to achieve if we choose performance as first factor with a weight close to one and if all others factors are extensive quantities such as densities or efficiencies.

3.2 A generalized parametric utility metric

With the factors achieved performance P, usable memory M, power consumption $Power$, space consumption $Space$, and peak performance $Peak$ as arbitrary scale, we hence arrive for the utility metric UM of a supercomputer (SC) at:

$$UM(SC) = P^{\alpha} \times \left(\frac{P}{Peak}\right)^{\beta} \times \left(\frac{M}{Peak}\right)^{\gamma} \times \left(\frac{Peak}{Power}\right)^{\delta} \times \left(\frac{Peak}{Space}\right)^{\epsilon} . \tag{3}$$

We have arranged our factors such that all weights are positive: $\alpha, \beta, \gamma, \delta, \epsilon \geq 0$.

Our scale Peak is arbitrary and UM must not depend on it. This requirement gives us a relation between the weights of our factors: $\beta + \gamma = \delta + \epsilon$. We can now use this relation to eliminate the arbitrary scale $Peak$ and one weight (β) and transform UM into:

$$UM(SC) = P^{\alpha} \times \left(\frac{M}{P}\right)^{\gamma} \times \left(\frac{P}{Power}\right)^{\delta+\epsilon} \times \left(\frac{Power}{Space}\right)^{\epsilon} . \tag{4}$$

We have chosen this form as it expresses UM as function of sustained performance, effective Byte/Flop ratio, power-efficiency, and power-density. These four very familiar quantities are often used in discussion of architectures and technologies (see Table 1).

It should be noted that δ is still the weight for the influence of power consumption on UM even if power-efficiency shows up in our expression with a weight of $\delta + \epsilon$. The reason is that the last factor power-density contains an additional contribution with weight ϵ from power consumption. We also believe that area is the appropriate measure of space requirements for supercomputers and not volume.

Table 1 Components of UM, their associated weights, and typical dimensions

Factor	Name	Typical dimension	Weight
P	Sustained Performance	$GFlop/s$	α
$\frac{M}{P}$	Effective Byte/Flop Ratio	$\frac{GByte}{GFlop/s}$	γ
$\frac{P}{Power}$	Power-efficiency	$\frac{GFlop/s}{kW}$	δ
$\frac{Power}{Space}$	Power-density	$\frac{kW}{m^2}$	ϵ

3.3 The influence of memory and power on UM

To illustrate the influence of memory and power weights in UM we consider the relative change in utility for a system upgrade where we keep performance constant. This implies that the values of α has no influence here and we will also ignore the influence of space in this section ($\epsilon = 0$). We illustrate the case of quadrupling memory and doubling power consumption for which we show the ratio $UM(1 \times P, 4 \times M, 2 \times Power)/UM(P, M, Power)$ in Fig. 1. We see that if we put less than roughly half the emphasis on memory compared to power $\gamma < 0.5 \times \delta$, the utility of the new system will be lower than before with a minimum of 0.5 if we put maximum weight on power and no value on memory. Otherwise

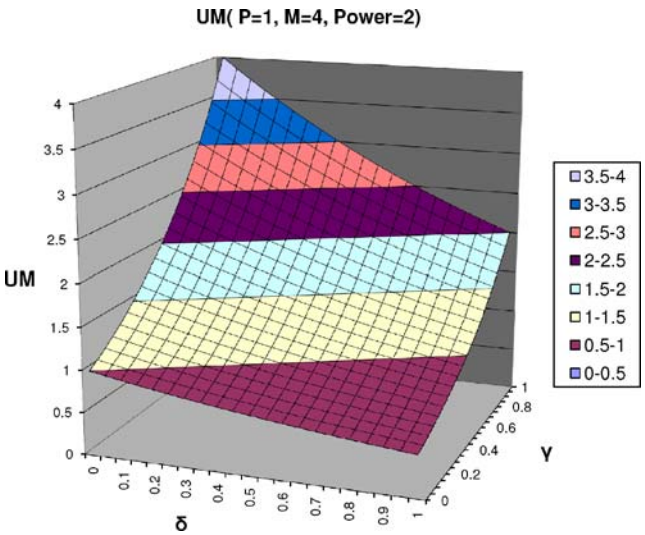

Fig. 1 The impact of the combined memory and power related factors in UM illustrated for an upgrade quadrupling memory and doubling power consumption. γ is the weight for memory and δ for power consumption

UM will grow rapidly with γ and has a maximum value of 4 limited by our increase of M for the case of maximum weight on memory and no weight on power.

The values shown in Fig. 1 are examples for the effective correction factors build into *UM*. They are the only reason that a ranking of systems based on *UM* would be different from a purely performance based ranking. Therefore they should serve as guide for selecting appropriate values for δ and γ.

3.4 Different scenarios for doubling utility

Equation 4 presents a fairly general class of utility metrics. To restrict the space of possible functions and hence the free weights further, we now have to answer the fundamental question we posed initially: *"When is computer A x-times as useful as computer B?"* To approach this question we assume four different answers and look at the resulting consequences for *UM*. We have summarized these four scenarios in Table 2.

A: As first answer (A) we assume that doubling an existing system doubles its utility: $UM(2 \times A) = 2 \times UM(A)$. This seems a natural answer for a situation where we have to choose the size of a system to buy. For *UM* it leaves us with only the restriction that the weight of performance $\alpha = 1$ as the value of all the densities in our formula do not change in this case. The remaining weights $(\delta, \epsilon, \gamma)$ can still be selected arbitrarily. *UM* increases linear with performance P even as power consumption increases linear. Any reduction of the increase in power consumption would result in an additional increase of *UM* above linear. In other words, this is a primarily performance based metric with correction factors for memory, power, and other features.

B: Now, if we want to replace an existing system with a new system, we might decide, that the new system is only twice as useful if it provides $2 \times P$ and $2 \times M$ in the same physical envelope (answer B). The resulting relation now couples some of our weights through the condition: $\alpha + \delta + \epsilon = 1$. Our emphasis on memory γ can still be selected independent. One should note that the answers A and B can not be combined meaningfully as they would imply $\delta + \epsilon = 0$, and thus for positive weights $\delta = \epsilon = 0$, which would remove any influence of power or space on *UM*. This relation also implies for positive δ and ϵ that $\alpha < 1$.

Table 2 Four different scenarios for what it means to double the utility of a computer system and the resulting conditions between the weights in *UM*

	P	M	Power, space	Resulting relation	k
A	$2 \times P$	$2 \times M$	$2 \times$	$\alpha = 1$	–
B	$2 \times P$	$2 \times M$	$1 \times$	$\alpha + \delta + \epsilon = 1$	1
C	$2 \times P$	$1 \times M$	$1 \times$	$\alpha + \delta + \epsilon - \gamma = 1$	0
D	$2 \times P$	$2^k \times M$	$1 \times$	$\alpha + \delta + \epsilon - (1-k)\gamma = 1$	k

C: If we have a fixed workload, which we wish to execute faster, we might consider a system with $2 \times P$ and otherwise unchanged characteristics as twice as useful (scenario C). This would be equivalent to a processor upgrade e.g. dual core to quad core in an existing system. The resulting equation now couples all four weights in: $\alpha + \delta + \epsilon - \gamma = 1$. We might try to combine this answer with answer A, but the resulting condition $\delta + \epsilon = \gamma$ would severely limit the available choices for the remaining weights. It would imply that our preference for memory is equal to our preference for the combined power and space efficiencies, which seems to be a very non-intuitive condition.

D: Based on the old saying, that job run-times are determined by job-queue limits, we might decide, that our new system has double the utility of our old one, if the largest problem fitting in memory executes at twice the previous performance and finishes in the same time as before (scenario D). For simplicity we assume that flop counts and memory requirements of our problem scale with simple exponents of an input parameter n called problem size. In case of the Linpack benchmark the number of operations scales with $O(n^3)$ while the required memory scales with $O(n^2)$. This implies that we have to scale the memory M in our system only with $P^{2/3}$. This result is easily generalized and we call this scaling exponent k. We recognize the resulting condition for our weights: $\alpha + \delta + \epsilon - (1-k)\gamma = 1$ as super-set of the conditions for scenarios B and C for the values $k = 1$ and $k = 0$ (see Table 2). Increasing *UM* linear with P is now only possible within the same physical envelope and with appropriate increases of memory.

Based on this we are left with A and D as the only two fundamentally different scenarios. For our naïve upgrade A we found the condition $\alpha = 1$, while for the problem scaling upgrade D we have: $\alpha = 1 - \delta + (1-k)\gamma - \epsilon$.

For simplicity reasons we will ignore for the remainder of this paper space requirements: $\epsilon = 0$. We will also refer to the position taken in answer D as "problem scaling utility metric" (UM_{ps}) with:

$$UM_{\mathrm{ps}}(SC) = P^{1-\delta+(1-k)\gamma} \times \left(\frac{M}{P}\right)^{\gamma} \times \left(\frac{P}{Power}\right)^{\delta}. \qquad (5)$$

3.5 System upgrade scenarios for UM_{ps}

We can now go back and consider the relative changes on UM_{ps} for upgrade scenarios which we summarize in Table 3. All these scenarios make the simplified assumption that the performance of our workload scales ideally and that we double effective performance with our upgrades. Less than ideal scaling will be considered in the next section.

A: If we consider the value of doubling an existing system (A) we see, that the utility increase depends on the trade-off of the memory (γ) and power (δ) weights and the problem

Table 3 The value of several different upgrades for the problem scaling utility UM_{ps} of Eq. 5

Scenario	P	M	Power	Change in UM_{ps}
A Doubling a system	$2 \times P$	$2 \times M$	$2 \times$ Power	$2^{1-\delta+(1-k)\gamma}$
B Replacing a system	$2 \times P$	$2 \times M$	$1 \times$ Power	$2^{1+(1-k)\gamma} \geq 2$
C Upgrading processor	$2 \times P$	$1 \times M$	$1 \times$ Power	$2^{1-k\gamma} \leq 2$

scaling exponent (k). Large emphasis on power will reduce the value of this upgrade while large emphasis of memory will increase its value as long as problem sizes grow slower than operation counts ($k < 1$).

B: Replacing a system by doubling performance and memory within the old system envelope and power (B) will always yield a utility increase larger than twice. This reflects the fact that doubling the memory allows executing problems larger than the scaled problem size, used in the definition of the utility metric. This represents added value and thus an increase in utility above the ratio of performance.

C: If we consider a power neutral upgrade of a processor (C) such as replacing a dual-core with a quad-core, the utility increase will always be less than twice. This is due to not changing the memory size and thus not being able to run a scaled problem size.

The analysis of these basic cases demonstrates that *UM* has many desirable properties, which a utility metric for computer systems should have.

3.6 Maximum values for *UM*

We now consider less than ideal scaling of performance of our workload in an upgrade of a system. Let $P(n)$ be the performance of system n times as "large" as the previous system. We set $\epsilon = 0$ in Eq. 4:

$$UM(n \times SC) = P(n)^{\alpha} \times \left(\frac{M(n)}{P(n)}\right)^{\gamma} \times \left(\frac{P(n)}{Power(n)}\right)^{\delta} . \quad (6)$$

For scenario A we substitute the increased memory $M(n) = n \times M$ and $Power(n) = n \times Power$ and transform the resulting expression into:

$$\begin{aligned} UM(n \times SC) &= \left(\frac{P(n)}{P}\right)^{\alpha-\gamma+\delta} \times n^{\gamma-\delta} \\ &\quad \times P^{\alpha} \times \left(\frac{M}{P}\right)^{\gamma} \times \left(\frac{P}{Power}\right)^{\delta} \\ &= S(n)^{\alpha-\gamma+\delta} \times n^{\gamma-\delta} \times UM(SC) . \end{aligned} \quad (7)$$

where $S(n)$ is the speedup of the larger system relative to the original one ($S(n) = P(n)/P$).

We now look for the system size to maximize *UM*. We differentiate Eq. 7 by n and transform the result into:

$$\begin{aligned} \frac{dUM(n \times SC)}{dn} &= \left((\alpha-\gamma+\delta)\frac{\frac{dS(n)}{dn}}{S(n)} + (\gamma-\delta)/n\right) \\ &\quad \times UM(n \times SC) . \end{aligned} \quad (8)$$

We set the expression of equation 8 to 0 and arrive at the condition:

$$(\alpha-\gamma+\delta) \times \frac{dS(n)}{dn} \times \frac{n}{S(n)} = \delta - \gamma . \quad (9)$$

Equivalent expressions for scenarios B and C can be derived and they differ from Eq. 9 only by the absence of δ and $\delta - \gamma$ on the right hand sides.

As long as our emphasis on performance α is larger than on power or memory the first factor on the left hand side will always be positive. We can then analyze the location of maximum *UM* by considering the different cases for the right

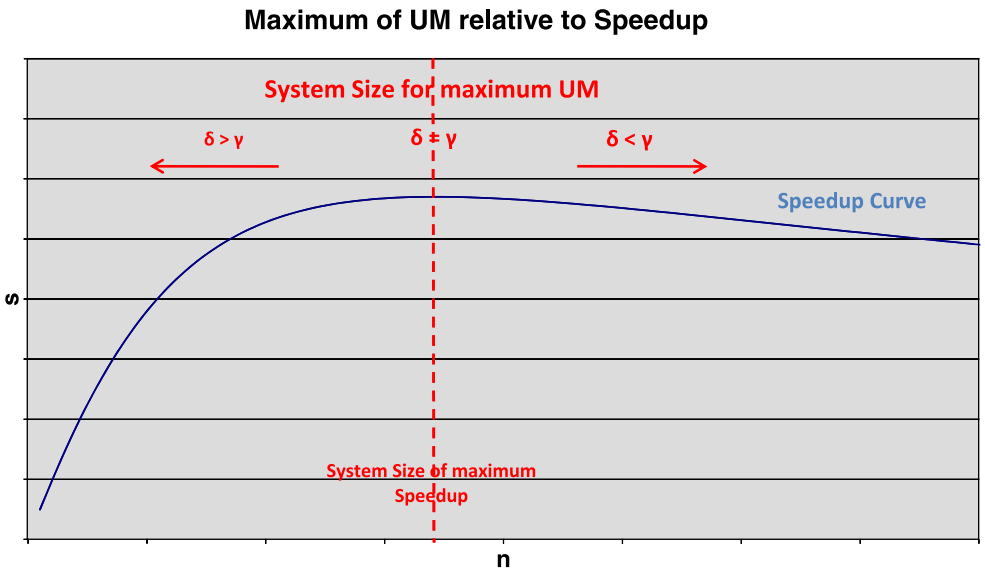

Fig. 2 The relative location of the maxima of the speedup curve $S(n)$ and the utility metric *UM* for different weights for memory γ and power consumption δ. Larger emphasis on memory shifts the maximum of *UM* to the right, larger emphasis on power to the left

hand side. The sign of $\delta - \gamma$ is now the same as of the slope of the speedup curve $dS(n)/dn$, which allows us to visualize the different cases in Fig. 2.

In case of $\delta = \gamma$ (or for scenario C), we conclude the location of the maximum of UM is determined by the derivative of the speedup by system size: $dS(n)/dn$. In other words $UM(n)$ raises and falls with $S(n)$ and both are maximal at the same n.

If we put larger emphasis on memory than on power, $\delta \leq \gamma$, (or for scenario B) the slope of the speedup curve for maximum UM will already be negative. The system size for maximum UM will therefore be shifted to larger system sizes n compared to the speedup curve (right half of Fig. 2). This is due to the comparably larger benefit of the memory increase. UM would still increase past the previous optimal size even as speedup already goes down.

For more emphasis on power than memory, $\delta \geq \gamma$, the slope of the speedup curve will still be positive for maximum UM . Thus the system size for maximum UM will be shifted to smaller system sizes n compared to the speedup curve, due to the larger benefit of conserving power (left half of Fig. 2). UM would start declining even before the maximum speedup is reached. Again we see that UM as utility metric for computer systems exhibits many properties we would expect.

4 Impact on the TOP500 rankings

To visualize the potential changes in rank order on the Top500, we took the subset of systems from the November 2008 edition for which we have actual measured power consumption data and problem size values N_{max}. For most cases power consumption was communicated to us by vendors of systems or individual sites. To approximate available user memory we used the square of recorded N_{max} values. We selected the problem scaling utility metric (UM_{ps}) and used the Linpack scaling parameter $k = 2/3$. Memory and power weights are set arbitrarily to $\gamma = \delta = 0.2$ for illustration. The effective α values is then computed with $1 - \delta + (1 - k)\gamma = 0.86667$.

Figure 3 shows Linpack and UM_{ps} values for our subset normalized to the respective value of the largest system. We recognize that UM_{ps} provides a correction to the Top500 ranking and that reordering would indeed take place. The amount of reordering is restricted by the size of the weight parameters chosen. For UM_{ps} substantially larger weights for memory and power would in turn reduce the weight applied to performance. This would reduce the extensive nature of UM_{ps} and for $\alpha \to 0$ would turn it into an intensive property with all the associated undesired effects.

To analyze the effects of the correction factors for memory and power we plot both factors in Fig. 4. The ratio between largest to smallest correction factors for memory is about 1.4 and for power is about 1.9. While these are sizable corrections an anti-correlation between the two factors for individual systems is also clearly visible and we calculated it to be -0.7. A possible interpretation is that systems with smaller memory consume less power. However, our example was meant for illustration only and sample selection for this analysis was most certainly biased. Therefore this interpretation should be considered with care.

5 Conclusions

The power consumption of supercomputers has become a major concern for the HPC users and industry. In most

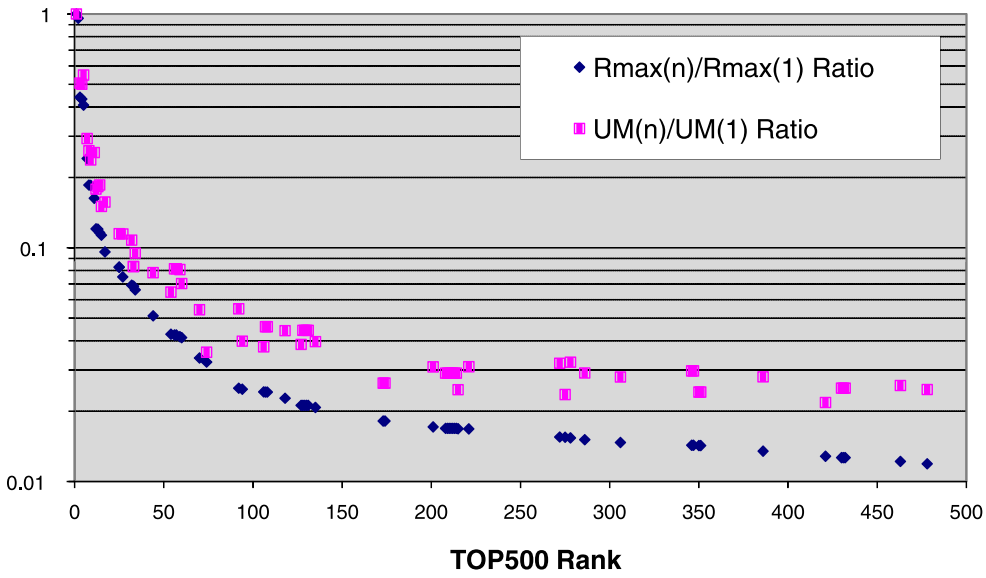

Fig. 3 Relative metric values for Linpack performance (R_{max}) and UM_{ps} over the Top500 rank which illustrates the amount of potential reordering

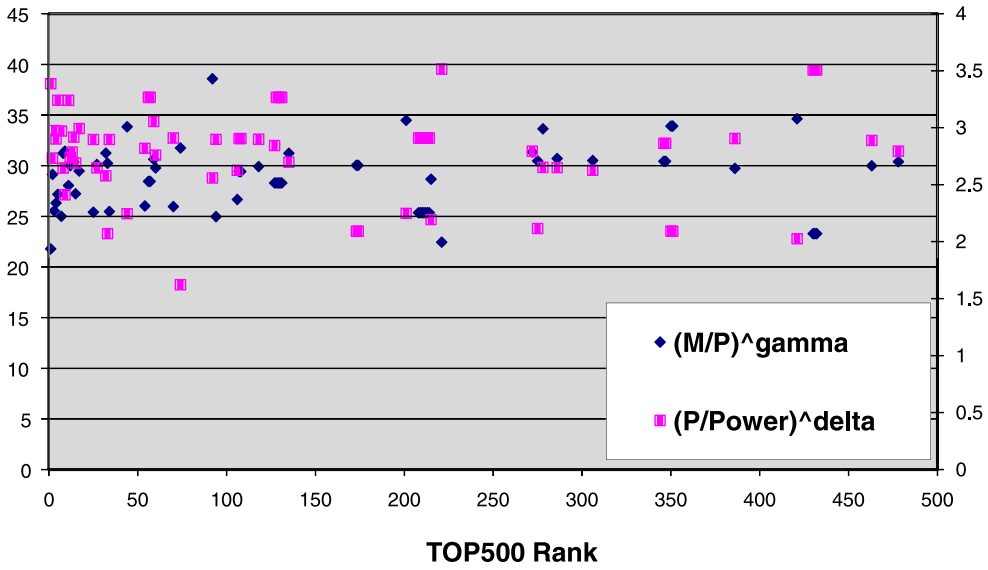

Fig. 4 Effective memory (*left axis*) and power (*right axis*) related correction factors in UM_{ps} for the Top500 systems. They show an anti-correlation off −0.7 which indicates that system with less memory in this sample use less energy

situations rankings of supercomputer systems ignore the effects of power consumption or similar system features. A prominent example is the Top500 project which ranks installed systems with the Linpack performance. Some recent attempts to improve this situation such as the Green500 have fundamental problems resulting from their use of intensive metrics such as power-efficiency for ranking individual computer systems. Intensive quantities such as efficiencies or densities are very useful to compare technologies but can not be used to rank objects by size.

In this paper we develop a new concept for a generalized parameterized utility metric (*UM*) to overcome previous limitations. We describe in detail how to incorporate any desired system feature in the metric (Eq. 3). In our framework a *UM* incorporating the effects of performance, memory, power, and space can be formulated naturally as a weighted product of performance, Byte to Flop ratio, power-efficiency, and power-density (Eq. 4). This is a very aesthetically pleasing result as all these quantities are commonly used in architecture and technology discussions.

Analyzing different answers to the question *"When is my new supercomputer twice as useful as my old one?"* we find relations between the free weights in *UM*. Based on these we define a problem scaling utility metric UM_{ps}, which increases linear with performance only if systems grow within the same physical envelope and with increases in memory appropriate for their workload (Eq. 5). Naïve doubling of systems will no longer yield twice the utility for this metric. This clearly demonstrates that *UM* incorporates the effects of power consumption and overcomes limitations of previous attempts to rank systems with performance and power related metrics.

This new metric can be used to redefine what a supercomputer is and help us to track how a realistic utility of our supercomputer systems truly improves over time. To illustrate this potential we calculated UM_{ps} values for the subset of systems in the Top500 for which power and memory are available and show the resulting changes in ranking.

References

1. Top500 supercomputer sites. http://www.top500.org (2009)
2. Strohmaier E, Dongarra J, Meuer HW, Simon HD (1999) The marketplace of high-performance computing. Parall Comput 25(13–14):1517–1544
3. Dongarra J, Luszczek P, Petitet A (2003) The LINPACK benchmark: past, present and future. Concurr Comput Pract Exper 15:1–18
4. The Green500 List. http://www.green500.org/ (2009)
5. HPC Challenge Benchmark. http://icl.cs.utk.edu/hpcc/ (2009)
6. SPEC: Standard performance evaluation corporation. http://www.spec.org (2009)
7. Feng W-C, Cameron K (2007) The Green500 List: Encouraging Sustainable Supercomputing. Computer 40(12):50–55
8. Makimoto T, Eguchi K, Yoneyama M (2001) The Cooler the Better: New Directions in the Nomadic Age. Computer 34(4): 38–42
9. STREAM: Sustainable memory bandwidth in high performance computers. http://www.cs.virginia.edu/stream (2009)
10. Oliker L, Carter J, Wehner M, Canning A, Ethier S, Mirin A, Parks D, Worley PH, Kitawaki S, Tsuda Y (2005) Leading Computational Methods on Scalar and Vector HEC Platforms. Proc SC 2005:62
11. Oliker L, Canning A, Carter J, Iancu C, Lijewski M, Kamil S, Shalf J, Shan H, Strohmaier E, Ethier S, Goodale T (2007) Scientific Application Performance on Candidate PetaScale Platforms. Proc IPDPS 2007:1–12

12. Luszczek P, Dongarra J, Koester D, Rabenseifner R, Lucas B, Kepner J, McCalpin J, Bailey D, Takahashi D (2005) Introduction the the HPC challenge benchmark suite. Available at http://www.hpcchallenge.org/pubs/

13. Kramer W, Shalf J, Strohmaier E (2005) The NERSC Sustained System Performance (SSP) Metric. Lawrence Berkeley National Laboratory. Paper LBNL-58868. http://repositories.cdlib.org/lbnl/LBNL-58868